IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	Dmitry Grebenev
Serial No.:	10/784,498
Filing Date:	February 23, 2004
Confirmation No.:	2208
Group Art Unit:	2113
Examiner:	Elmira Mehrmanesh
Title:	**KERNEL-LEVEL METHOD OF FLAGGING PROBLEMS IN APPLICATIONS**

MAIL STOP: Amendment

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Dear Sir:

DECLARATION PURSUANT TO 37 C.F.R. § 1.131

I, the undersigned, hereby declare and state that:

1. I am over the age of 21 years, of sound mind, and competent in all respects to make this Declaration.

2. I am the inventor of the subject matter of the above-referenced patent application, entitled *Kernel-Level Method of Flagging Problems in Applications*, filed on February 23, 2004 (the "Application").

3. Prior to February 13, 2001, I gained a full understanding of the subject matter of at least the current version of Claims 1-20 of the Application (the "Invention"); therefore, I conceived the subject matter of the Invention prior to February 13, 2001.

4. Beginning after conception of the Invention, I participated in the design and creation of a software program that incorporated the subject matter of the Invention (the "Software") and that was complete prior to February 13, 2001; therefore, I reduced the subject matter of the Invention to practice prior to February 13, 2001.

5. Prior to July 11, 2002, I confidentially used the Invention, internal to Computer Associates (the "Assignee of the Invention"), to research and test future products that were, then, under development by the Assignee of the Invention.

6. All of the work in conceiving and reducing the Invention to practice occurred in the United States.

7. Attached as Exhibit A is a copy of a document that includes source code excerpts from the Software. The document, portions of which have been redacted for privacy reasons, existed prior to February 13, 2001 and is provided as evidence that the Software was reduced to practice prior to February 13, 2001. The source code excerpts from the Software that are attached as Exhibit A include instructions regarding all of the functionality included in Claims 1-20.

8. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true. Further, I declare that these statements are made with the knowledge that willful false statements, and the like so made, are punishable by fine or imprisonment, or both, under Section 1001, Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the Application or any patent issuing thereon.

Declaration pursuant to 37 C.F.R. § 1.131 in regard to 10/784,498.

Signed this 19 day of January, 2007.

D. Grebenev

Dmitry Grebenev

EXHIBIT A

SOURCE CODE

[redacted]

[redacted]

[redacted]

[redacted]

```
#!/usr/bin/perl
##############################################################################
# static char _SCCS_Id[] = "@(#)memleaks.pl      1.7
# static char _CA_Copyright[] = "Copyright (C) [redacted]
International, Inc.  All rights reserved.";
##############################################################################
;# This script will try to identify any processes that may have a memory
;# leak by gathering kernel-level data using q4, then analyzing it
;# Prerequisites: an HP-UX box with q4 (that would be any HP-UX 10.x and
higher)
;#    -- gredm02, [redacted]
;#
;#    1.2    gredm02  Added pid to a process name to distinguish multiple copies
;#                    Added mail notifcation to myself when finished
;# 1.3-4     gredm02  Corrected tne number of points (n+1) for n intervals
;#    1.5    gredm02  Added conditional definition for LoadMatchingProcesses sub
;# 1.6-7     gredm02  Added a requirement to use Perl v.5 and later
;#

($THIS = $0) =~ s|.*/(\w+)|\1|g ; # script name to use in messages

$GOODPERL = 5.0 ;
die "Please upgrade to Perl ver.$GOODPERL or higher to run this script.\n".
    "We have ready-to-run binaries for most platforms.\nYou can ask platform
owners for directions.\n\n" unless $] >= $GOODPERL ;

;# Command-line options, e.g.,
;#  -t90    collect data every 90 seconds
;#  -n50    collect data for 50 intervals of $interval each

;# Parse command-line options
foreach $i (@ARGV)
{
      ($switch = $i) =~ s|-([a-z])\d+|\1| ;
      ($value = $i) =~ s|-[a-z](\d+)|\1| ;
      if ($switch eq 'n')
      {
          $n = $value ;
      }
      elsif ($switch eq 't')
      {
        if ($value < 180) {
            print STDERR "$value seconds is too low for the interval -
resetting to 180\n" ;
            $interval = 180 ;
        } else {
            $interval = $value ;
        }
      }
      else
      {
       print STDERR "usage: $THIS [-n#] [-t#]\n\a" ;
```

[redacted]

[redacted]

```
            exit(1) ;
        }
}

;# Provide defaults if values are not supplied on the command line
if (! defined($interval))  {
   $interval = 600 ; # check every 600 seconds
}

if (! defined($n)) {
   $n = 10 ;   # gather data for 10 intervals of $interval seconds each
}

$n = $n + 1 ;  #  N points make (N-1) intervals

;# Sanity checks
if (! -x "/usr/bin/q4" && ! -x "/usr/contrib/bin/q4") {
      print STDERR "Cannot locate q4. Exiting\n\a" ;
      exit (2) ;
}

print "Gathering data for $n intervals, $interval seconds each...\n" ;

;# Globals
$USER_TO_BE_NOTIFIED = 'gredm02@cai.com' ;  # change to what you want
$period = 0 ; # interval counter
$Q4CMDFILE = "/tmp/q4.commands" ;
$DATAFILE = "/tmp/mydata" ;
$PRIMER   = "/tmp/q4.primer" ;
$RESULTS  = "/tmp/myresult" ;
chop($THIS_BOX = `uname -n`) ;        # for mail notification and such

open (Q4CMDS, ">$Q4CMDFILE") || die("write $Q4CMDFILE\n") ;
print Q4CMDS <<'EOC';
#
# This is a Perl script for q4. It uses q4 library functions
#

&Include("misc.pl") ;
&Include("processes.pl") ;
&Include("commands.pl") ;

;#
;#  Conditional definitions for releases that don't have them
;#
if (! defined(&LoadAllProcs)) {
   sub LoadAllProcs
   {
      return &Load("-r struct proc from proc max nproc");
   }
}

if (! defined(&LoadMatchingProcesses)) {
  sub LoadMatchingProcesses
  {
    # Create a pile consisting of the processes that match the
    # stated criterion.
```

[redacted]

[redacted]

```
    local($criterion) = $_[0];
    local($N);

    &LoadAllProcs();          # get them all
    $N = &Keep($criterion); # keep the ones we want
    &Forget(-1);              # clean up (toss the previous pile)

    return $N;                # return how many we got
  }
}

sub PregionData
{
     &PushHistory ;
     &Keep("p_pid == $pid") ;
     ($name, $vas) = &GetFields("p_comm  p_vas") ;
     local($rc0) = &Load("struct vas from p_vas") ;
     if ($rc0 > 0)
     {
        $i = 0 ;
        while ($i++ < 100)
        {
           $rc1 = 0 ;
           $rc2 = &Load("struct p_lle from addrof") ;  # vas
           $rc1 = &Load("struct pregion from lle_prev") if $rc2 ;
           if ($rc1 > 0)
           {
              local($ptype, $pregion) = &GetFields("p_type addrof") ;
              if ($ptype =~/PT_DATA/) # PT_DATA
              {
                 local($pregion, $data_pgcnt) = &GetFields("addrof p_count") ;
                 # Now we have all the data we want - spit it out ...
                 print STDOUT  "!X%  $name  $pid  $data_pgcnt\n" ;
                 last ;
              }
           }
        }
     }
     &PopHistory ;
}

sub main
{
  $rc = &LoadMatchingProcesses("p_pid > 100 && p_stat!=0") ;
  if ($rc == 0) {
     return ;
  }

  local(@pidlist) = &Get1Field("p_pid") ;
  foreach $pid (@pidlist)
  {
     # Get data from the process pregion
     &PregionData ;
     # Get data from ???
  }
  close OUT ;
}
```

[redacted]

[redacted]

```
EOC

close Q4CMDS ;

unlink $DATAFILE ;

open (PRIMER, ">$PRIMER") || die("cannot create primer file\n") ;
print PRIMER "include $Q4CMDFILE \nrun main\n" ;
close (PRIMER) ;

while($period++ < $n) {
      print "$THIS: Gathering data for period $period ...\n" ;
    chop($timestamp = `date "+%m/%d/%y > %H:%M"`) ;
    open(OUT, ">>$DATAFILE") || die("cannot append to $DATAFILE\n") ;
      print OUT "------  $period: $timestamp  -------\n" ;
    close OUT ;
    system("q4 -p /stand/vmunix /dev/mem <$PRIMER 2>/dev/null >>$DATAFILE") ;
    if ($rc) {
          print STDERR "$THIS: q4 errors while processing $Q4CMDFILE\n" ;
          exit (2) ;
      }
      sleep($interval) ;
}

;# The following part is logically independent and can be made into a
;# separate script.
;# Now process the data we have collected using q4

print STDERR "$THIS: Now processing data gathered in the $DATAFILE file...\n"
;
open(F, "$DATAFILE") || die("open $DATAFILE\n") ;
$period = 0 ;
while(<F>)
{
   if (/^-----/) {
       $period++ ;
       next ;
   }
   next unless /!X%/ ;
   $_ = ' ' . $_ ;
   ($junk, $name, $pid, $size) = split ;
   $when = sprintf("%03d", $period) ;
   $proc{join(':',$name.'.'.$pid, $when)} = $size ;
}
close F ;

open(OUT2, ">$RESULTS") || die("write $RESULTS file\n") ;
foreach $ii (sort (keys %proc))
{
      #($name, $period) =~ m|(A-Za-z)(\d\d*)| ;
      print OUT2  $ii,":",$proc{$ii},"\n" ;
}
close OUT2 ;

#
#  Below the grow variable is incremented if the process grew from
#  the previous check, while "count" counts check intervals.
#  The goal is to identify processes which grew during each check interval,
```

[redacted]

[redacted]

```
#  i.e., when we process it we get grow == count
#
open(IN2, $RESULTS) || die("read $RESULTS file\n") ;
while (<IN2>)
{
        chop ;
        ($name, $period, $size) = split(/:/) ;
        if ($prevname eq $name) {
                $count++ ;
                if ($lastsize < $size + 0) {
                        $grow++ ;
                }
        } else {
                if ($grow >  1) # $count /2)
          {
                   print "$prevname grew in ", $grow, " out of ", $count, "
checks\n" ;
                }
                $prevname = $name ;
                $count = 0 ; # reset
                $grow = 0 ;
        }
        $lastsize = $size ;
}
print STDERR "$THIS: completed successfully.\n" ;
system("echo memleaks.pl completed on $THIS_BOX | mail ".
$USER_TO_BE_NOTIFIED) ;
```

[redacted]

[redacted]